Heuristics and Exact Algorithms for Capacitated and Uncapacitated Choice-Based Pricing

Tom Haering [*] [†] Robin Legault[‡] Fabian Torres [*]
Michel Bierlaire [*]

May 8, 2025

Report TRANSP-OR 250508
Transport and Mobility Laboratory
School of Architecture, Civil and Environmental Engineering
Ecole Polytechnique Fédérale de Lausanne
transp-or.epfl.ch

[*]École Polytechnique Fédérale de Lausanne (EPFL), School of Architecture, Civil and Environmental Engineering (ENAC), Transport and Mobility Laboratory, Switzerland, {tom.haering, fabian.torres, michel.bierlaire}@epfl.ch

[†]Corresponding author.

[‡]Massachusetts Institute of Technology (MIT), Operations Research Center, USA, legault@mit.edu

Abstract

The choice-based pricing problem (CPP) consists in optimizing product prices while accounting for consumer preferences and potential capacity constraints. Demand is typically modeled using a discrete choice model (DCM). We introduce the Breakpoint Heuristic Algorithm (BHA) to address the CPP with and without capacity constraints as well as an extension of the Breakpoint Exact Algorithm (BEA) to handle capacities, together with valid inequalities for the QCQP-L (quadratically constrained quadratic program with linear objective) formulation of the uncapacitated CPP, allowing us to speed up the exact Branch & Benders Decomposition (B&BD) approach. Capacity management is handled by an exogenous priority queue. Results show that, in the capacitated case, the BEA solves a larger set of instances within the time limit than the state-of-the-art MILP formulation, achieving identical revenues on all completed instances and better solutions in cases where the MILP fails to converge. The proposed BHA heuristic performs well across both low- and high-dimensional instances, consistently producing near-optimal solutions with an average gap below 0.2%. In the uncapacitated case, BHA and its ILS extension solve all tested instances—including high-dimensional ones—for which exact methods such as BEA and B&BD often exceed time limits. The BHA also improves the performance of B&BD when used to guide the search and generate valid inequalities. In mixed-logit pricing problems, both BHA and ILS solve benchmark instances significantly faster than methods specialized for this setting, while maintaining solution quality; the ILS matches all known optima, and the BHA maintains an average gap below 0.02%.
Keywords: discrete choice, pricing, capacity constraints, heuristic, valid inequalities

Glossary

Acronym	Full term
CPP	Choice-based Pricing Problem
DCM	Discrete Choice Model
ML	Mixed Logit
MILP	Mixed Integer Linear Programming / Program
QCQP-L	Quadratically Constrained Quadratic Program with Linear objective
BEA	Breakpoint Exact Algorithm
BHA	Breakpoint Heuristic Algorithm
ILS	Iterated Local Search heuristic
B&B	spatial Branch and Bound
B&BD	spatial Branch and Benders Decomposition
CoBiT	Convexification of a Biconvex optimization and Trust-region algorithm
LAG	Lagrangian decomposition-based heuristic

1 Introduction

Effective pricing strategies play a critical role in various industries, especially in markets where customer preferences and product choices are diverse and complex. Discrete choice models (DCMs), based on random utility theory, provide a robust framework for capturing the impact of customer heterogeneity on demand. By modeling how individuals select among available alternatives, these models allow companies to better predict demand and optimize their pricing strategies accordingly. Building on this foundation, we aim to address the challenge of developing efficient algorithms for pricing with disaggregate demand modeling in the form of a DCM, which we will refer to as the choice-based pricing problem (CPP). Our central research question is how to create a scalable and computationally efficient algorithm to solve the CPP, especially for high-dimensional instances with customer heterogeneity and when taking constraints on capacity into account.

By advancing state-of-the-art methods, we seek to provide more effective solutions for real-world pricing challenges, where customer preferences and demand patterns can be modeled using random utility maximization (RUM) frameworks, resulting in so-called choice-based optimization problems. Utility maximization theory posits that individuals assess each option available to them and choose the one that maximizes their utility. Key areas where these models are applied include pricing Davis et al. (2017); Gallego & Wang (2014); Li et al. (2019); Paneque et al. (2022) and assortment optimization Liu et al. (2020); Rusmevichientong et al. (2010), which are fundamental to strategic business decisions. DCMs like logit and nested logit are a widely used tool to model demand in various problem settings, including facility location (Mai & Lodi, 2020; Ljubić & Moreno, 2018), railway timetabling (Cordone & Redaelli, 2011; Robenek et al., 2018), and toll setting (Wu et al., 2012) problems, as well as revenue management and pricing (Shen & Su, 2007; Korfmann, 2018; Gallego & Wang, 2014; Müller et al., 2021). For the latter, however, it has been shown that the more complex mixed logit (ML) DCMs represent the more powerful and realistic demand representation (Sumida et al., 2021; van de Geer & den Boer, 2022; Marandi & Lurkin, 2023). Incorporating advanced discrete choice models such as mixed logit into optimization problems introduces significant computational difficulties due to the non-convex nature of the choice probabilities (Hanson & Martin, 1996) and the need to represent individual demand perspectives, such as capacity allocation. As a result, only small to moderate-sized problems are typically solvable to optimality (Benati & Hansen, 2002; Paneque et al., 2021).

To tackle these computational challenges, innovative approaches have emerged: Gilbert et al. (2014) developed a tractable approximation for maximizing revenue through pricing under mixed logit (ML) demand in congested networks, utilizing a two-step process combining a mixed integer program with an ascent algorithm. Subsequently, Li et al. (2019) explored a price optimization problem under discrete ML demand, introducing a pair of concave maximization problems to bound the revenue function.

It is important to highlight that in Li et al. (2019) and van de Geer & den Boer (2022), both of which are closely related to our work, the probability measure is considered discrete. In Li et al. (2019), customer-specific variables are not included in the utility models, re-

sulting in identical choice probabilities for all customers. In contrast, van de Geer & den Boer (2022) connects only the exogenous component of the utility to the customers. Consequently, unlike van de Geer & den Boer (2022), more recent studies, as well as our framework, address customer heterogeneity, including variations in price sensitivity parameters. For instance, Marandi & Lurkin (2023) consider discrete ML pricing with heterogenous price-sensitivity parameters and propose an iterative optimization algorithm that asymptotically converges to the optimal solution. By formulating a linear optimization problem based on the trust-region approach, they find a feasible solution and design a convex optimization problem using a convexification technique to approximate the optimization problem from above. A branching method is then used to tighten the optimality gap. Results show that for various tested instances (up to 5 customers and 5 alternatives), their method significantly outperforms other approaches, among which van de Geer & den Boer (2022), in almost all cases. They furthermore successfully demonstrate the benefits in terms of expected revenue when going from MNL to ML demand modeling, a finding later confirmed by for example Abdolhamidi & Lurkin (2024)

One way to deal with the non-convexity arising from advanced DCMs is simulation-based optimization (Gosavi, 2015). This framework involves optimizing stochastic simulations to derive effective decisions or strategies. For instance, in choice-based optimization, this method approximates demand for products or services by simulating various random choices based on utility functions and using a sample average approximation method Haase & Müller (2013); Legault & Frejinger (2024).

To provide a more general framework for integrating advanced choice models into optimization problems, Paneque et al. (2021) proposed a simulation-based approach to formulate any choice-based optimization problem as a mixed-integer linear program (MILP). While increasing complexity due to the exponential scaling of the MILP solve time with the number of draws, this approach guarantees convergence to globally optimal solutions for sufficiently large numbers of draws. However, its practical applicability is limited to small-scale instances, highlighting the need for more efficient computational strategies. The authors focus on the choice-based pricing problem (CPP), where a DCM is integrated to model the demand in a pricing optimization problem, with the goal of maximizing profit. They manage to solve instances with two controlled prices, 50 customers and 250 simulation draws to optimality within two hours for the uncapacitated case and within 21 hours when taking capacity into account.

Recognizing the issue of scale, Paneque et al. (2022) introduced a heuristic method based on a Lagrangian decomposition scheme, where the original problem is simplified by grouping simulation scenarios based on their similarities, allowing for a more efficient and scalable approach to solving the aforementioned MILP formulation. However, the efficiency of the algorithm depends heavily on the scenario grouping strategy, and its effectiveness might vary across different setups or datasets. They use the same data set as Paneque et al. (2021) and focus on the CPP with capacity constraints. For two controlled alternatives and 50 individuals, they manage to solve instances with 100 draws up to around 3% optimality within a two-hour time limit. For four controlled alternatives they solve instances with 25 draws in the same time limit to around 3.5% optimality. It is

worth mentioning, however, that adding capacity constraints drastically complicates the optimization problem.
Haering et al. (2023) introduce two exact methods for the uncapacitated CPP, the Breakpoint Exact Algorithm (BEA), as well as a Spatial Branch and Benders Decomposition (B&BD) approach. With the BEA, they solve instances with one controlled price, 50 customers and one million simulation draws within less than two minutes to optimality, as well as instances with two controlled prices, 50 customers and 22 thousand draws within 72 hours. With the B&BD algorithm, they manage to solve instances with four controlled prices, 50 customers and 200 draws to 1% optimality within 24 hours. Comparing to mixed-logit specific algorithms for the CPP, they demonstrate that they outperform the work from Marandi & Lurkin (2023) by a factor of 300x for two-price optimization using the BEA and an average factor of 3x for larger numbers of alternatives when using the spatial Branch and Bound (B&B) algorithm.
These results show that the BEA and B&B / B&BD approaches are efficient methods for solving the CPP without capacity constraints. However, the curse of dimensionality is a strong limiting factor even for the Branch and Bound approaches, and none of their algorithms are capable of handling capacity constraints.

To summarize, we find that for the CPP, there are no solution approaches in the literature that are at the same time general, flexible, and capable of solving realistic instances in a reasonable amount of time, especially for high numbers of prices and when constraints on capacity are introduced. Our aim is to address this gap by extending the BEA algorithm to handle capacity constraints, developing an efficient and flexible heuristic for high-dimensional problems, the Breakpoint Heuristic Algorithm (BHA), and enhancing the capabilities of the B&B and B&BD procedures by adding valid inequalities. The BHA exploits the fact that the BEA performs exceptionally well for one-dimensional instances but scales exponentially for more prices. Our coordinate descent approach optimally exploits this characteristic. Furthermore, we show how the solution from the BHA can be used to speed up the exact spatial B&BD approach for the non-capacitated problem by guiding the pruning and introducing valid inequalities for the QCQP-L formulation of the problem, which the B&BD is based on. The goal in this context is to additionally demonstrate that a simulation-based method with a larger number of samples can be solved more efficiently than an approach that directly employs nonlinear choice functions but is restricted to a smaller number of classes or breaking points, as shown in Legault & Frejinger (2024). This provides a strong motivation for using a purely simulation-based approach with deterministic customers rather than approximating ML models with a smaller sample of customers following a logit model. By using a larger number of samples, the simulation-based method can more accurately capture the diversity of customer preferences and behaviors, thereby enhancing the robustness of the results and providing a more comprehensive understanding of complex customer heterogeneity.
The paper is structured as follows: Section 2 describes the choice-based pricing problem and its formulation as a mathematical program. In Section 3, we introduce the Breakpoint Heuristic Algorithm (BHA), as well as an extension with an iterated local search (ILS) to escape local optima. In Section 4 we demonstrate how to guide the B&BD algorithm

using the solution from the BHA together with new valid inequalities, followed by Section 5 presenting the computational experiments. Finally, we conclude the paper and present its essential takeaways in Section 6.

2 Problem definition

Consider a competitive market with multiple products, of which J products are controlled by a supplier who wants to identify the set of prices that maximizes their revenue. We number the controlled alternatives from 1 to J, and the competitors' alternatives using non-positive numbers, from $1-K$ to 0. Denote $C^1 = \{1, \dots, J\}$ the set of all offered alternatives by the supplier and $C = C^1 \cup \{1-K, \dots, 0\}$ the set of all alternatives available. We consider N customers choosing one product among all offered alternatives. Each individual $n \in \mathcal{N} = \{1, \dots, N\}$ may furthermore have a different set of considered alternatives, denoted by $C_n \subset C$. We need to assume that each individual has at least one uncontrolled alternative in the choice set. If not, the problem is unbounded. An individual's considered set of alternatives that are offered by the supplier is denoted by $C_n^1 = \{i \in C_n | i \in C^1\}$. The behavior of the customers is captured by a random utility model: each alternative $i \in C_n$ is associated with a stochastic utility U_{in}, which depends on socioeconomic characteristics of individual n, alternative-specific attributes, and the controlled prices for alternatives $i \geq 1$. It can be defined as follows:

$$\begin{aligned} U_{in} &= V_{in} + \varepsilon_{in} && \forall i \in C_n \setminus C_n^1, \\ U_{in} &= V_{in} + \beta_p^{in} p_i + \varepsilon_{in} && \forall i \in C_n^1, \end{aligned}$$

where V_{in} represents the deterministic part of the utility that is observed by the analyst, which can take any form and be non-linear in the explanatory variables, and ε_{in} is the unobserved error term (and thus a random variable). It is furthermore worth noting that the price variable p_i can be separated into multiple p_{ic} for different customer segments c. For the sake of readability, we adhere to the global price notation p_i. The probability $P_n(i)$ that individual n chooses alternative $i \in C_n$ can now be written as follows:

$$P_n(i) = \mathbb{P}(U_{in} \geq U_{jn} \, \forall j \in C_n) \qquad \forall i \in C_n$$

The controlled prices $p_i, i \in C^1$ are decision variables that need to be optimized in order to maximize the expected revenue, expressed as each product's price times the probability the product is bought by an individual, summed up over all individuals. We assume each price p_i to be bounded within a continuous domain $[p_i^L, p_i^U]$. In general, the mathematical expression for $P_n(i)$ is complex. Advanced models, such as mixtures of logit and hybrid choice models lack a closed form and are expressed using integrals (Hanson & Martin, 1996).

2.1 Problem formulation

To address the lack of closed-form expressions for the probability functions, we employ the simulation approach of Paneque et al. (2021): We take R draws ε_{inr} from the distri-

bution of the error terms to generate R scenarios (the terms "scenario" and "draw" will henceforth be used interchangeably) with deterministic utilities U_{inr}:

$$\begin{aligned} U_{inr} &= V_{in} + \beta_p^{in} p_i + \varepsilon_{inr} && \forall i \in C^1, n \in \mathcal{N}, r \in \mathcal{R}, \\ &= c_{inr} + \beta_p^{in} p_i && \forall i \in C^1, n \in \mathcal{N}, r \in \mathcal{R}, \\ U_{inr} &= V_{in} + \varepsilon_{inr} && \forall i \in C \setminus C^1, n \in \mathcal{N}, r \in \mathcal{R}, \\ &= c_{inr} && \forall i \in C \setminus C^1, n \in \mathcal{N}, r \in \mathcal{R}, \end{aligned}$$

where $\mathcal{R} = \{1, \dots, R\}$ and c_{inr} contains all terms of the utility function independent from the price. As now all uncontrolled alternatives $i \in C \setminus C^1$ have a utility that is constant given an individual n and a scenario r, we can gather them for each tuple (n, r) as a single opt-out alternative, corresponding to the best of them:

$$c_{0nr} = \max_{i \in C \setminus C^1} c_{inr} \qquad \forall n \in \mathcal{N}, r \in \mathcal{R}.$$

We thus redefine $C = C^1 \cup \{0\}$ and impose $0 \in C_n \ \forall n \in \mathcal{N}$, as otherwise, the problem is unbounded. The choice of individual n in scenario r is then modeled with the choice variable ω_{inr}, which is equal to 1 if alternative i is chosen and 0 otherwise. Subsequently, the probability of an individual $n \in \mathcal{N}$ choosing alternative $i \in C_n$ can be approximated by the sample average $\frac{1}{R} \sum_{r \in \mathcal{R}} \omega_{inr}$.

This framework leads to the formulation of the uncapacitated choice-based pricing problem (CPP) as a quadratically constrained quadratic program with linear objective (QCQP-L), presented in Haering et al. (2023), given in Formulation 1.

The objective function is equal to the expected revenue and is thus defined as the approximated choice probability of individual n selecting alternative $i \in C_n$ multiplied by the alternative's price p_i, summed over all individuals. The constraints define the individual choices: Constraints (μ_{nr}) guarantee that exactly one alternative is chosen per individual and scenario. Constraints (κ_{inr}) model the utility U_{inr} of each alternative i for individual n in scenario r. Constraints (ζ_{nr}) and constraints (α_{inr}) enforce the optimality conditions for the customer utility maximization problem. Note that this is a continuous, but non-convex, reformulation of the original mixed-integer linear program (MILP) approach. Indeed the previously integer ω variables are relaxed to be in $[0, 1]$ instead, at the cost of non-convex choice constraints. More precisely, the utility maximization problem at the individual level resembles a knapsack problem, where the choice variable of the most attractive alternative is forced to take the value 1. However, the inclusion of an endogenous variable in the "weights" of the knapsack transforms the problem from linear to bilinear, as it involves the product of two continuous variables: the price p_i and the choice variable ω_{inr}, as isolated in the constraints (λ_{inr}). Finally, the price variables may also depend on the individuals (or groups of individuals), thus allowing for segmented targeting of the population.

Formulation 1 can be extended to incorporate capacity constraints on the alternatives, as demonstrated in Paneque et al. (2021). However, adding these constraints makes it no longer possible to relax the domain of the ω variables, meaning the problem has to be

$$\max_{p,\omega,\eta,U,h} \frac{1}{R} \sum_{r\in\mathcal{R}} \sum_{n\in\mathcal{N}} \sum_{i\in C_n^1} \eta_{inr}$$

s.t.

$$\begin{aligned}
&\sum_{i\in C_n} \omega_{inr} = 1 && \forall n \in \mathcal{N}, r \in \mathcal{R} \quad (\mu_{nr}) \\
&h_{nr} = c_{0nr}\omega_{0nr} \\
&\qquad + \sum_{i\in C_n^1} [c_{inr}\omega_{inr} + \beta_p^{in}\eta_{inr}] && \forall n \in \mathcal{N}, r \in \mathcal{R} \quad (\zeta_{nr}) \\
&h_{nr} \geq c_{0nr} && \forall n \in \mathcal{N}, r \in \mathcal{R} \quad (\alpha_{0nr}) \\
&h_{nr} \geq U_{inr} && \forall n \in \mathcal{N}, i \in C_n^1, r \in \mathcal{R} \quad (\alpha_{inr}) \\
&U_{inr} = c_{inr} + \beta_p^{in} p_i && \forall n \in \mathcal{N}, i \in C_n^1, r \in \mathcal{R} \quad (\kappa_{inr}) \\
&\eta_{inr} = p_i \omega_{inr} && \forall n \in \mathcal{N}, i \in C_n^1, r \in \mathcal{R} \quad (\lambda_{inr}) \\
&\omega \in [0,1]^{(J+1)NR} \\
&p \in [p_1^L, p_1^U] \times \ldots \times [p_J^L, p_J^U] \\
&\eta \in [0, p_1^U] \times \ldots \times [0, p_J^U] \\
&U, h \in \mathbb{R}^{JNR}, \mathbb{R}^{NR}
\end{aligned} \tag{1}$$

Formulation 1: QCQP-L model for the uncapacitated CPP

$$\max_{p,\omega,\eta,U,h} \frac{1}{R} \sum_{r\in\mathcal{R}} \sum_{n\in\mathcal{N}} \sum_{i\in C_n^1} \eta_{inr}$$

s.t.

$$\begin{aligned}
&\sum_{i\in C_n} \omega_{inr} = 1 && \forall n \in \mathcal{N}, r \in \mathcal{R}, && (\mu_{nr})\\
&h_{nr} \le z_{inr} + M(1-\omega_{inr}) && \forall n \in \mathcal{N}, i \in C_n, r \in \mathcal{R}, && (\zeta_{nr})\\
&h_{nr} \ge z_{inr} && \forall n \in \mathcal{N}, i \in C_n, r \in \mathcal{R}, && (\alpha_{inr})\\
&U_{inr} = c_{inr} + \beta_p^{in} p_i && \forall n \in \mathcal{N}, i \in C_n^1, r \in \mathcal{R}, && (\kappa_{inr})\\
&\eta_{inr} \le \omega_{inr} p_i^U && \forall n \in \mathcal{N}, i \in C_n^1, r \in \mathcal{R}, && (\lambda_{inr}^1)\\
&\eta_{inr} \le p_i && \forall n \in \mathcal{N}, i \in C_n^1, r \in \mathcal{R}, && (\lambda_{inr}^2)\\
&\eta_{inr} \ge p_i - (1-\omega_{inr}) p_i^U && \forall n \in \mathcal{N}, i \in C_n^1, r \in \mathcal{R}, && (\lambda_{inr}^3)\\
&z_{0nr} = c_{0nr} && \forall n \in \mathcal{N}, r \in \mathcal{R}, && (a_{nr})\\
&z_{inr} \le U_{inr} && \forall n \in \mathcal{N}, i \in C_n^1, r \in \mathcal{R}, && (b_{inr})\\
&z_{inr} \ge U_{inr} - M(1-y_{inr}) && \forall n \in \mathcal{N}, i \in C_n^1, r \in \mathcal{R}, && (c_{inr})\\
&z_{inr} \le M/2 + M y_{inr} && \forall n \in \mathcal{N}, i \in C_n^1, r \in \mathcal{R}, && (d_{inr})\\
&\omega_{inr} \le y_{inr} && \forall n \in \mathcal{N}, i \in C_n^1, r \in \mathcal{R}, && (e_{inr})\\
&\sum_{m=1}^{n} \omega_{imr} \le (c_i - 1) y_{inr} && && \\
&\qquad + (n-1)(1-y_{inr}) && \forall n > c_i \in \mathcal{N}, i \in C_n^1, r \in \mathcal{R}, && (f_{inr})\\
&\sum_{m=1}^{n} \omega_{imr} \ge c_i (1-y_{inr}) && \forall n > 1 \in \mathcal{N}, i \in C_n^1, r \in \mathcal{R}, && (g_{inr})
\end{aligned} \tag{2}$$

$$\omega \in \{0,1\}^{(J+1)NR},$$
$$y \in \{0,1\}^{JNR},$$
$$p \in [p_1^L, p_1^U] \times \ldots \times [p_J^L, p_J^U],$$
$$\eta \in [0, p_1^U] \times \ldots \times [0, p_J^U],$$
$$U, z, h \in \mathbb{R}^{JNR}, \mathbb{R}^{(J+1)NR}, \mathbb{R}^{NR}.$$

Formulation 2: MILP model for the capacitated CPP

written as a MILP. Furthermore, some constraints presented in Formulation 1 have to be adjusted. For the sake of clarity, we present the full MILP formulation for the CPP with capacity constraints, as it represents the state-of-the-art to model and solve the problem. First, we assume a set of given capacities $c_i, i \in C^1$. We now introduce two new sets of variables, the binary y_{inr} variables, which indicate the availability of alternative $i \in C^1$ for individual n in scenario r, and the discounted utility variables z_{inr}, which are equal to the utilities U_{inr} if alternative $i \in C^1$ is available (i.e. $y_{inr} = 1$) and set to a low enough value otherwise, thus embedding the concept of unavailable alternatives in the customer subproblem. We note that individuals are assumed to be numbered in the order of their priority to access the market. This numbering can follow any rule, including being random, but must be exogenous, as without it, the solver would be allowed to endogenously prioritize customers to maximize overall revenue, leading to unrealistic and biased outcomes. In many applications (such as ticket sales, online bookings, or real-world markets), customers arrive or are served according to an exogenous order that the supplier cannot control. Simply imposing a total capacity constraint like $\sum_{n \in \mathcal{N}} y_{inr} \leq c_i$ would eliminate the sequential nature of access and would substantially alter the modeled problem.

Lastly, as the choice variables are now again binary, the product $p_i \omega_{inr}$ can be modeled in a linear way using big-M constraints, where the optimal big M is the largest possible value taken by said product, i.e. p_i^U. The full MILP formulation for the capacity-constrained CPP is given in Formulation 2, where M is a large enough constant. Constraints (ζ_{nr}) and (α_{inr}) have been adjusted to incorporate the newly added z variables. The non-convex constraints (λ_{inr}) have been replaced by a set of linear constraints (λ^1_{inr}), (λ^2_{inr}) and (λ^3_{inr}), modeling the big-M linearization of the aforementioned product. Constraints $(a_{inr} - d_{inr})$ define the discounted utility z_{inr}, ensuring that in case of $y_{inr} = 0$, z_{inr} takes on a small enough value to not compete with any available alternative's utility, and is set to be equal to U_{inr} otherwise. Constraint (e_{inr}) enforces that an alternative can only be chosen if it is available to that individual in that scenario. Lastly, constraints (f_{inr}) and (g_{inr}) define the y variables: Constraint (f_{inr}) sets y_{inr} to 0 whenever the capacity of alternative i is reached and constraint (g_{inr}) makes sure that, if an alternative is no longer available, there have to be enough people choosing the alternative to fill it.

It is furthermore worth noting that, if the prices are all fixed to constant values, it is trivial to find the optimal values of all variables. To see this, it is enough to remember that with fixed prices p, the problem reduces to solving a utility maximization knapsack problem, this time with constant weights, for each customer n and scenario r. As the prices are the only connecting variables, this can be done separately for every tuple (n, r), giving an efficient way to evaluate the objective function for a given feasible solution. The same procedure can be applied to the capacitated version of the problem, taking a priority queue into account.

3 Methodology

In this section, we extend the BEA to incorporate capacity constraints, introduce the Breakpoint Heuristic Algorithm (BHA), which can be applied to solve both the uncapacitated and capacitated version of the CPP, as well as an iterated local search (ILS) heuristic to improve the solution quality. Lastly, we demonstrate how to guide exact methods using the heuristic result together with newly developed valid inequalities for the CPP.

3.1 Extending the BEA for capacity constraints

The Breakpoint Exact Algorithm (BEA) is an efficient exact method for solving the Choice-based Pricing Problem (CPP) when the number of alternatives is small. It systematically explores all local optima by enumerating candidate solutions over structured subsets of the feasible space. As shown in Algorithm 1, the BEA iterates over all $J!$ permutations of the ordering of price variables. We denote by S the set of all possible permutations s of $\{1, \ldots, J\}$. For a given permutation $s \in S$, the j^{th} element of the ordered list s in denoted by s_j. For each ordering, the algorithm initializes the price vector and customer utilities, then calls a recursive procedure `enumerate` that incrementally sets prices according to the given order. At each step, customer choices and accumulated revenues are tracked, and the best solution across all permutations is retained. The algorithm's worst-case time complexity is $O(J!(NR)^J \log(NR))$, which is exponential in the number of alternatives J, but polynomial in the number of individuals N and simulation draws R.

The recursive function `enumerate` constructs solutions by fixing one price at a time and solving the resulting subproblem. A key feature is that, at the final level of recursion—when the last alternative is introduced—the profit can be dynamically updated without recomputing customer utility maximizations from scratch. Thanks to the assumption of non-decreasing price ordering, previously computed customer decisions remain valid unless the new alternative becomes more attractive. This allows for fast, incremental updates to the total revenue by tracking customer switches, leading to significant computational savings.

Algorithm 1: Breakpoint Exact Algorithm (BEA) to solve the CPP

Result: optimal solution p^* and objective value o^* for Formulation 1.
$p^*_j \leftarrow 0 \quad \forall j \in \{1, \ldots, J\}$;
$o^* \leftarrow 0$;
for s **in** S **do**
  $p_{s_j} \leftarrow 0 \quad \forall j \in \{1, \ldots, J\}$;
  $h^{s_1}_{nr} \leftarrow c_{0nr} \quad \forall (n, r) \in \mathcal{N} \times \mathcal{R}$;
  $\bar{\eta}_{nr} \leftarrow 0 \quad \forall (n, r) \in \mathcal{N} \times \mathcal{R}$;
  $(\hat{p}, \hat{o}) \leftarrow$ `enumerate`$(s, p, h^{s_1}, \bar{\eta}, 1)$;
  if $\hat{o} > o^*$ **then**
    $p^* \leftarrow \hat{p}$;
    $o^* \leftarrow \hat{o}$;
  end
end
return (p^*, o^*);

To incorporate capacity constraints into the BEA, we employ a streamlined variant that systematically explores each valid combination of breakpoints in sequence. As for the uncapacitated case, the optimal price ensures that the utility of a product matches the utility of the next cheapest alternative for at least one customer and scenario, maximizing revenue without unnecessary customer loss. Specifically, for an optimal price $p_i, i \in C^1$, there exists a customer $n \in \mathcal{N}$ and scenario $r \in \mathcal{R}$ such that any increase in p_i by $\varepsilon > 0$ would lower the utility U_{inr} below that of cheaper alternatives or the opt-out option, deterring that customer and decreasing overall revenue. Hence, this price acts as a "breakpoint" or "indifference point" in the customer's decision-making, representing the maximum price before their interest shifts to more affordable options.

The outer level algorithm (see Algorithm 3) remains the same as for the BEA, with the only difference being that the recursive `enumerate` function is replaced by a function `enumerate_cap`, described in Algorithm 4. At its deepest level, it invokes the function `compute_objective_value` that calculates the objective value for a set price variables fixed to a combination of breakpoints, taking capacity restrictions into account. Due to interdependent choices potentially leading to recursive substitution, continuously updating choices and revenues becomes impractical. Additionally, when adding a new product, calculating breakpoints for each simulated customer from their previous preference to the new option is insufficient in this problem setting. Instead, breakpoints must be computed from any possible previous product to the new one, as capacity limits may force customers to choose an alternative other than their most preferred. This adjustment accounts for decision breakpoints involving switches from any introduced product to the new one. The process of sequentially introducing alternatives remains the same as in the original BEA, as the order in which alternatives are introduced to customers alters their decision making breakpoints. A diagram visualizing the BEA with capacity constraints is shown in Figure 1.

These two changes, compared to the BEA without capacity constraints, increase the algo-

rithm's computational complexity. However, directly evaluating the objective function at each breakpoint combination also enhances flexibility in revenue computation methods. Indeed, this allows to add any type of constraint to the problem without adjusting the algorithm. In our case, we only consider capacity constraints, implemented with an exogenous priority queue. Algorithm 2 lays out the evaluation of revenue given such a queue, where individuals are assigned the highest utility alternative with positive remaining capacity.

Algorithm 2: Compute objective value with priority queue

Function `compute_objective_value`$(p, c, \texttt{prio\_queue})$**:**
- $\sigma \leftarrow (0)_{i \in C}$
- **for** $idx \in \texttt{prio\_queue}$ **do**
  - $u \leftarrow [U^i_{idx}$ for $i \in C]$
  - $a \leftarrow \text{sort}(u, \text{descending})$
  - $\varphi \leftarrow$ false
  - $j \leftarrow 1$
  - **while** $j \leq C - 1$ **and** $!\varphi$ **do**
    - **if** $\sigma_{a_j} \leq c_{a_j} - 1$ **then**
      - $\sigma_{a_j} \mathrel{+}= 1$
      - $\varphi \leftarrow$ true
    - **end**
    - **else**
      - $j \mathrel{+}= 1$
    - **end**
  - **end**
- **end**
- $o \leftarrow \sum_{i \in C} \sigma_i \cdot p_i$
- **return** o

end

Algorithm 3: Breakpoint exact algorithm (BEA) to solve the capacitated CPP

Result: optimal solution p^* and objective value o^* for the capacitated CPP.

$p^*_j \leftarrow 0 \quad \forall j \in \{1, \ldots, J\}$

$o^* \leftarrow 0$

for s **in** S **do**
- $p_{s_j} \leftarrow 0 \quad \forall j \in \{1, \ldots, J\}$
- $(\hat{p}, \hat{o}) \leftarrow \texttt{enumerate\_cap}(s, p, 1)$
- **if** $\hat{o} > o^*$ **then**
  - $p^* \leftarrow \hat{p}$;
  - $o^* \leftarrow \hat{o}$;
- **end**

end

return (p^*, o^*)

Algorithm 3, like Algorithm 1 iterates over all possible orderings of prices $p_{s_1} \leq p_{s_2} \leq$

$\cdots \leq p_{s_j}$, $s \in S$. Each restricted problem is addressed by the recursive `enumerate_cap` function. This function accepts as arguments the current permutation $s \in S$ of alternatives, a partially filled vector of prices p, with components $p_{s_1} \leq \cdots \leq p_{s_{j-1}}$ already set, and the depth j of the current permutation's exploration.
The function `enumerate_cap` solves the capacitated CPP, restricted to a specific ordering of prices, in a recursive manner.

Algorithm 4: Recursive enumeration function within the BEA, when taking capacities into account

Function `enumerate_cap(s, p, j)`**:**
- $\bar{p}_{s_j}^{nrs_i} \leftarrow \frac{u_{nr}^{s_i} - c_{s_j nr}}{\beta_p^{s_j n}} \quad \forall (n, r) \in \mathcal{N} \times \mathcal{R}, i < j \in C \cup \{0\}$
- $\mathcal{N}_2 \leftarrow \{(n, r, s_i) | p_{s_j}^L < \bar{p}_{s_j}^{nrs_i} < p_{s_j}^U\}$
- $\mathcal{N}_2 \leftarrow \mathcal{N}_2 \cup \{p_{s_j}^L, p_{s_j}^U \; \forall i \in C\}$
- Sort the elements of $\mathcal{N}_2$ from largest to smallest
- **if** $j \leq J - 1$ **then**
  - **for** $\bar{p}_{s_j}^{nrs_i} \in \mathcal{N}_2$ **do**
    - $p_{s_j} \leftarrow \bar{p}_{s_j}^{nrs_i}$
    - $(\hat{p}, \hat{o}) \leftarrow$ `enumerate_cap(s, p, j + 1)`
    - **if** $\hat{o} > o^*$ **then**
      - $o^* \leftarrow \hat{o}$
      - $p^* \leftarrow \hat{p}$
    - **end**
  - **end**
- **end**
- **else**
  - **for** $\bar{p}_{s_j}^{nrs_i} \in \mathcal{N}_2$ **do**
    - $p_{s_j} \leftarrow \bar{p}_{s_j}^{nrs_i}$
    - $o \leftarrow$ `compute_objective_value(p)`
    - **if** $o > o^*$ **then**
      - $o^* \leftarrow o$
      - $p^* \leftarrow p$
    - **end**
  - **end**
  - **return** (p^*, o^*)
- **end**

end

The exponential growth in BEA's complexity with the number of controlled alternatives J is illustrated in the tree diagram in Figure 1, where each branch represents a call to the recursive `enumerate_cap` function.

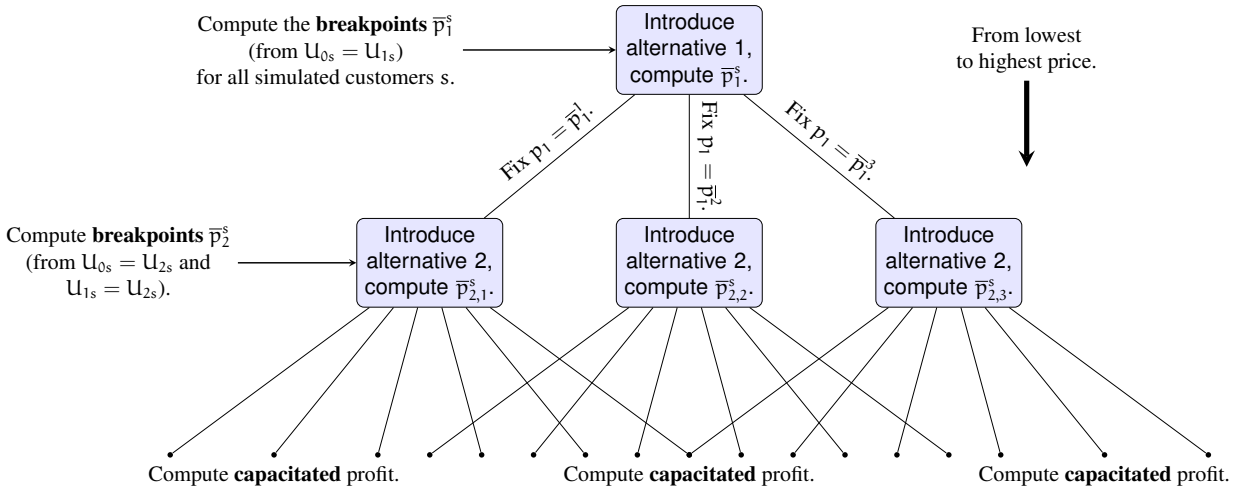

Figure 1 – BEA with capacity constraints for three simulated customers and two alternatives.

3.2 Breakpoint heuristic algorithm (BHA)

The BHA is based on the exact Breakpoint Exact Algorithm (BEA) and capitalizes on the idea of decision-making breakpoints. Given a fixed state of the market—a set of controlled and competing alternatives, all with fixed prices—we can compute, for each customer and each simulation scenario, a breakpoint at which the utility of a newly introduced alternative becomes the largest, thus altering the current decision. These breakpoints represent a set of local optima that can be enumerated. The algorithm's complexity is $O(J!(NR)^J \log(NR))$, and thus polynomial in N and R, but exponential in the number of prices.

The BHA can be summarized as a coordinate descent (ascent), iteratively optimizing one price at a time while fixing all other prices, terminating once no coordinate can be improved further. This approach leverages the fact that the BEA can solve instances involving a single price variable very efficiently, while its computational time increases exponentially with the number of price variables. By iteratively treating all but one price as fixed, the BHA is able to navigate the search space more effectively and maintain computational traceability. It is described in the following procedure:

1. Choose a starting point for the heuristic. As any combination of prices is feasible, the simplest choice here can be to choose the middle of the price bounds, $p^* = (\frac{p_i^L + p_i^U}{2})_{i \in C^1}$.
2. Evaluate the objective function for price p^*, giving objective value o^*.
3. Set $j = 1$.

4. Solve the problem using the BEA but with modified bounds $\bar{p}^L, \bar{p}^U$, where, $\bar{p}_i^L = \bar{p}_i^U = p_i \; \forall i \neq j$ and $\bar{p}_j^L = p_j^L$, $\bar{p}_j^U = p_j^U$. We refer to these new bounds as $\bar{p}^L(p, j, p^L)$ and $\bar{p}^U(p, j, p^U)$. All bounds, except for alternative j, are thus tight, greatly simplifying the problem. We thus iterate over all relevant breakpoints for all simulated customers, evaluating the objective value at each combination of breakpoints and updating the highest objective o^* and the best prices p^* whenever a better solution is found.

5. Set $j = j + 1$ and repeat from step 4. In the case of $j = D$, we reset it to $j = 1$.

6. Terminate once no change in the optimal solution is observed over D iterations.

The pseudocode for the BHA is provided in Algorithm 5.

Algorithm 5: Breakpoint Heuristic Algorithm (BHA)

```
Function BHA(p_start; c, prio_queue):
    o* ← compute_objective_value(p_start)
    p* ← p_start
    p ← p*
    j ← 1
    σ ← 0
    while σ < D do
        p̂_j, ô ← BEA(p̄^L(p, j, p^L), p̄^U(p, j, p^U); c, prio_queue)
        p_j ← p̂_j
        if ô > o* then
            o* ← ô
            p* ← p̂
            σ ← 0
        else
            σ += 1
        end
        j += 1
        if j > D then
            j ← 1
        end
    end
    return o*, p*
end
```

3.3 Iterated local search heuristic (ILS)

The iterated local search algorithm is an iterative enhancement to the BHA, aimed at escaping local optima through adaptive step size adjustments. Initiated with a set of initial prices p and an objective value o^*, the algorithm takes as additional inputs an initial step size δ, the number of steps to be taken in each direction (increase and decrease) k, a step increase factor γ, and a maximum step size Δ_{max}. Each iteration consists of the following steps:

- Exploring both increase and decrease directions for each price component $i \in C^1$, the algorithm tests k increment steps, each being a multiple of δ. At each step, a new candidate solution $\bar{p}$ is generated by adjusting the i-th component of p by $\pm k\delta$.
- For each candidate solution, the objective function is evaluated, and o^* is updated when a new best solution is found.
- If there was no improvement in objective value after completing the line search on all components of p, the step size δ is increased by the factor γ, enlarging the scope of the line search in the next iteration.

The algorithm continues until $\delta \geq \Delta_{\max}$. The pseudocode for the ILS can be found in Algorithm 6.

Algorithm 6: Iterated Local Search Algorithm (ILS)

```
Function iterated_local_search(p_start, δ, k, γ, Δ_max; caps, prio_queue):
    o* ← compute_objective_value(p_start)
    p* ← p_start
    φ ← true
    σ ← 0
    while δ < Δ_max do
        φ ← false
        for j ∈ 1 : D do
            for d ∈ [−1, 1] do
                for l ∈ 1 : k do
                    p^new ← p
                    p_j^new += d · l · δ
                    if p_j^L ≤ p_j^new ≤ p_j^U then
                        o^new, p^new, ← BHA(p_new; c, prio_queue)
                        if o^new > o* and p^new ∈ [p^L, p^U] then
                            o* ← o^new
                            p* ← p^new
                            φ ← true
                            σ ← 0
                        end
                    end
                end
            end
        end
        if !φ then
            σ += 1
            δ ·= γ
        end
        else
            σ ← 0
        end
        p ← p*
        o ← o*
    end
    return o*, p*
end
```

4 Guiding the exact algorithm

As shown in Section 5, the heuristics introduced above provide very good solutions in a short amount of time. However, they do not provide a guarantee of optimality. Therefore, we also investigate the possibility of solving the problem exactly. In this context, we use the heuristic to help an exact algorithm. We consider as a starting point a spatial Branch

& Benders algorithm for the uncapacitated CPP. We give a brief overview of the method and then explain how we can speed up its convergence. The main strategy for accelerating the performance of Branch and Bound algorithms hinges on improving the bounds. Specifically, using a heuristic such as the BHA to find an initial feasible solution generates a strong lower bound (for a maximization problem) on the objective, significantly reducing the number of nodes explored in the search tree by enabling more effective pruning. Additionally, the upper bounds, needed to prove optimality, can be improved through the incorporation of valid inequalities. Lastly, we incorporate knowledge of the heuristic solution into the enumeration strategy to break ties between nodes with equal upper bound.

4.1 Spatial Branch and Benders algorithm

The spatial Branch and Benders (B&BD) algorithm solves the mathematical program stated in Formulation 1 by first employing the McCormick envelope (McCormick, 1976) to relax the bilinear constraints defining η_{inr}, and then tightens that relaxation by finding the best set of bound via a spatial branch and bound tree. In each child node, the length of the interval between the price bounds for a selected price is halved, guaranteeing convergence. The relaxation in each node is then solved using Benders decomposition.
Given a set of bounds $p_i \in [p_i^L, p_i^U]\ \forall i \in C^1$, the McCormick envelop used to relax the constraints $\eta_{inr} = p_i \omega_{inr}$ is given by:

$$\begin{aligned}
\eta_{inr} &\geq p_i^L \omega_{inr} \\
\eta_{inr} &\geq p_i^U \omega_{inr} + p_i - p_i^U \\
\eta_{inr} &\leq p_i^L \omega_{inr} + p_i - p_i^L \\
\eta_{inr} &\leq p_i^U \omega_{inr}
\end{aligned}$$

This yields the linear McCormick relaxation of the QCQP-L shown in Formulation 3. It is worth noting that the constraint (λ^1_{inr}), i.e., $\eta_{inr} \geq p_i^L \omega_{inr}$, is theoretically redundant given the bounded domain of p_i and the presence of the other McCormick envelope constraints. However, we observed empirically that its inclusion improves solver performance in several instances, likely due to enhanced propagation of variable bounds during presolve. For this reason, we have retained it in the formulation. Furthermore, constraint (λ^3_{inr}) can be replaced with the simpler bound $\eta_{inr} \leq p_i$ without loss of tightness. This holds because the constraint (λ^4_{inr}), which enforces $\eta_{inr} \leq p_i^U \omega_{inr}$, already ensures that $\eta_{inr} \leq 0$ when $\omega_{inr} = 0$. Thus, when combined with (λ^4_{inr}), the proposed constraint $\eta_{inr} \leq p_i$ is equivalent to (λ^3_{inr}) in all relevant cases. Although this is equivalent, keeping (λ^3_{inr}) in its current form does not alter the feasible region and may be beneficial for solver symmetry and structure.
A spatial Branch and Bound algorithm, see for example Liberti (2008), is employed to find the globally optimal values for all the prices. A conceptual outline of the method is given below:

1. Solve the McCormick relaxation (Formulation 3) using the initial bounds.

2. From the solution value of the prices, compute the corresponding choices and construct a feasible solution.

3. Choose a price to branch on, then split the search interval for that price, i.e., its bounds, into two, while all other price bounds remain the same. Add two new nodes to the Branch and Bound tree, each corresponding to one set of the new bounds.

4. Choose the next node from the tree based on the achieved objective value in its parent node (*best-first-search*), and solve the relaxation with the bounds corresponding to that node.

5. Continue until the relative gap between the objective value of the tightest relaxation is close enough (up to a predefined relative optimality gap) to the objective value of the best feasible solution found.

In every node of the Branch and Bound tree, Formulation 3 needs to be solved with a given set of price bounds, which may be time-consuming due to a large number of variables η and ω. However, Formulation 3 is highly separable: indeed, if all variables p_i are fixed to a certain value, the utility maximization problem can be solved for every individual and scenario independently. This is why a Benders decomposition approach is considered to speed up the solution of the McCormick relaxation in each node of the Branch and Bound tree. Benders decomposition works by decomposing the original problem into a master problem and a subproblem, where the master problem is a relaxation of the original problem that iteratively is improved by the addition of optimality and feasibility cuts, see for example Rahmaniani et al. (2017).

$$
\begin{aligned}
&\max_{p,\omega,\eta,U,h} \frac{1}{R} \sum_{r\in\mathcal{R}} \sum_{n\in\mathcal{N}} \sum_{i\in C_n^1} \eta_{inr} \\
&\text{s.t.} \\
&\sum_{i\in C_n} \omega_{inr} = 1 && \forall n \in \mathcal{N}, r \in \mathcal{R} \quad (\mu_{nr}) \\
&h_{nr} = c_{0nr}\omega_{0nr} \\
&\qquad + \sum_{i\in C_n^1} [c_{inr}\omega_{inr} + \beta_p^{in}\eta_{inr}] && \forall n \in \mathcal{N}, r \in \mathcal{R} \quad (\zeta_{nr}) \\
&h_{nr} \geq c_{0nr} && \forall n \in \mathcal{N}, r \in \mathcal{R} \quad (\alpha_{0nr}) \\
&h_{nr} \geq U_{inr} && \forall n \in \mathcal{N}, i \in C_n^1, r \in \mathcal{R} \quad (\alpha_{inr}) \\
&U_{inr} = c_{inr} + \beta_p^{in} p_i && \forall n \in \mathcal{N}, i \in C_n^1, r \in \mathcal{R} \quad (\kappa_{inr}) \\
&\eta_{inr} \geq p_i^L \omega_{inr} && \forall n \in \mathcal{N}, i \in C_n^1, r \in \mathcal{R} \quad (\lambda_{inr}^1) \\
&\eta_{inr} \geq p_i^U \omega_{inr} + p_i - p_i^U && \forall n \in \mathcal{N}, i \in C_n^1, r \in \mathcal{R} \quad (\lambda_{inr}^2) \\
&\eta_{inr} \leq p_i^L \omega_{inr} + p_i - p_i^L && \forall n \in \mathcal{N}, i \in C_n^1, r \in \mathcal{R} \quad (\lambda_{inr}^3) \\
&\eta_{inr} \leq p_i^U \omega_{inr} && \forall n \in \mathcal{N}, i \in C_n^1, r \in \mathcal{R} \quad (\lambda_{inr}^4) \\
&\omega \in [0,1]^{(J+1)NR} \\
&p \in [p_1^L, p_1^U] \times \ldots \times [p_J^L, p_J^U] \\
&\eta, U, h \in \mathbb{R}^{JNR}, \mathbb{R}^{JNR}, \mathbb{R}^{NR}
\end{aligned}
\tag{3}
$$

4.2 Valid inequalities

A first general set of valid inequalities can be derived by observing that, given a set of bounds $[p_i^L, p_i^U]_{i\in C^1}$, i.e. at each node in the tree, for each simulated customer $(n, r) \in \mathcal{N} \times \mathcal{R}$ and alternative $i \in C_n^1$, there exists a *minimal breakpoint* $\check{p}_i^{nr}$ (assuming strongest competition) and a *maximal breakpoint* $\hat{p}_i^{nr}$ (assuming weakest competition), defined as follows:

$$
\begin{aligned}
\check{p}_i^{nr} &= \frac{\max_{j\in C_n^1\setminus\{i\}} U_{jnr}(p_j^L) - c_{inr}}{\beta_p^{in}} && \forall n \in \mathcal{N}, i \in C_n^1, r \in \mathcal{R}, \\
\hat{p}_i^{nr} &= \frac{\max_{j\in C_n^1\setminus\{i\}} U_{jnr}(p_j^U) - c_{inr}}{\beta_p^{in}} && \forall n \in \mathcal{N}, i \in C_n^1, r \in \mathcal{R},
\end{aligned}
$$

where the $U_{inr}(p) = c_{inr} + \beta_p^{in} p \; \forall n \in \mathcal{N}, i \in C_n^1, r \in \mathcal{R}$ are seen as functions in the price p. For the strongest competition, we compute the maximum utility among all other controlled alternatives, given their lowest possible prices given the current bounds. As we assume $\beta_p < 0$ in all cases, this corresponds to the strongest possible competition. On the other hand, taking the maximum utility among all other controlled alternatives, given their

highest possible prices corresponds to the breakpoint assuming the weakest competition. These breakpoints exhibit the following properties:

$$\begin{aligned} p_i \;\leq\; \check{p}_i^{nr} &\implies (n,r) \text{ is guaranteed to select } i \\ &\implies \omega_{inr} \geq 1 \quad \forall n \in \mathcal{N}, i \in C_n^1, r \in \mathcal{R}, \\ p_i \;\geq\; \hat{p}_i^{nr} &\implies (n,r) \text{ is guaranteed to } \textit{not} \text{ select } i \\ &\implies \omega_{inr} \leq 0, \eta_{inr} \leq 0 \quad \forall n \in \mathcal{N}, i \in C_n^1, r \in \mathcal{R}, \end{aligned}$$

which can be integrated into Formulation 1 by use of the following valid inequalities:

$$p_i + (\check{p}_i^{nr} - p_i^L)\omega_{inr} \geq \check{p}_i^{nr} \qquad \forall n \in \mathcal{N}, i \in C_n^1, r \in \mathcal{R},$$

and

$$\begin{aligned} p_i + (p_i^U - \hat{p}_i^{nr})\omega_{inr} \leq p_i^U \qquad & \forall n \in \mathcal{N}, i \in C_n^1, r \in \mathcal{R}, \\ p_i + (p_i^U - \hat{p}_i^{nr})\eta_{inr} \leq p_i^U \qquad & \forall n \in \mathcal{N}, i \in C_n^1, r \in \mathcal{R}, \end{aligned}$$

respectively. If $p_i \leq \check{p}_i^{nr}$ and $\omega_{inr} = 0$ we have that $p_i \leq \check{p}_i^{nr}$ and $p_i \geq \check{p}_i^{nr}$ has to hold at the same time, which is a contradiction, thus $\omega_{inr} = 1$ is enforced. In the same way, if $p_i \geq \hat{p}_i^{nr}$ then ω_{inr} has to be set to 0 by the solver, as otherwise:

$$p_i + (p_i^U - \hat{p}_i^{nr}) \leq p_i^U \implies p_i - \hat{p}_i^{nr} \leq 0 \implies p_i \leq \hat{p}_i^{nr},$$

which again leads to a contradiction. The inequality for η_{inr} is derived in the same way. Note that in case of the upper bound with $\check{p}_i^{nr}$, we cannot deduce a bound on the revenue η_{inr} except that it is less than or equal to $\check{p}_i^{nr}$ which is already covered by the McCormick constraints.
Given a feasible solution p^* to the CPP, we can furthermore derive the following relations:

$$\begin{aligned} p_j \geq p_j^* \quad \forall j \neq i, \quad p_i \leq p_i^* \qquad &\implies \omega_{inr} \geq \omega_{inr}^* \qquad \forall n \in \mathcal{N}, i \in C_n^1, r \in \mathcal{R} \\ p_j \leq p_j^* \quad \forall j \neq i, \quad p_i \geq p_i^* \qquad &\implies \omega_{inr} \leq \omega_{inr}^* \qquad \forall n \in \mathcal{N}, i \in C_n^1, r \in \mathcal{R} \end{aligned}$$

We describe how to convert the first relation into linear constraints, as the second follows symmetrically. As the left-hand side consists of J individual conditions that all need to be verified, we introduce an auxiliary variable $z_{inr} \in [0,1]$ for each such condition i and simulated customer (n,r). Note that these variables are indicators for whether or not a condition holds, and as such are binary in nature. However, including them as binary variables would inevitably slow down the computation of the relaxation significantly, which is why we relax their domain. For each $i \in C^1$, we model the z variables with the following constraints:

Constraints for alternative i:

$$p_i + (p_i^* - p_i^L)z_{inr} \geq p_i^* \quad \forall n \in \mathcal{N}, r \in \mathcal{R}$$

Constraints for alternatives $j \neq i$:

$$p_j - (p_j^U - p_j^*)z_{jnr} \leq p_j^* \quad \forall n \in \mathcal{N}, r \in \mathcal{R}$$

Additionally, we add constraints to ensure the relationship between the z variables and the choice variables ω:

$$\omega_{inr} \geq \sum_{j=1}^{J} z_{jnr} - (J-1) \quad \forall n \in \mathcal{N}, r \in \mathcal{R}$$

These constraints ensure that $\omega_{inr} \geq 1$ if all conditions hold, i.e., $z_{inr} = 1 \; \forall i \in C_n^1$. Although the relaxation of the z variables' domain makes these inequalities less tight, they remain valid, as each individual inequality still holds for fractional values of z. However, this change makes the computational gain of their addition to the relaxation dependent on the problem instance.

The valid inequalities developed in this subsection can directly be integrated in Formulation 3.

4.3 Improving price bounds

Given a set of price bounds $[p_i^L, p_i^U]_{i \in C^1}$ at a node in the branch and bound tree, we can look to improve the price bounds further before solving the relaxation. For this, we consider the lowest minimal breakpoint $\check{p}_i$ and highest maximal breakpoint $\hat{p}_i$ for each $i \in C^1$ over all individuals and scenarios:

$$\check{p}_i \coloneqq \min_{n \in \mathcal{N}, r \in \mathcal{R}} \check{p}_i^{nr} \;\; \forall i \in C^1$$

$$\hat{p}_i \coloneqq \max_{n \in \mathcal{N}, r \in \mathcal{R}} \hat{p}_i^{nr} \;\; \forall i \in C^1$$

From their derivation, we can infer certain conditions based on these values. If $p_i > \hat{p}_i$, this implies that no customer will choose alternative i. Conversely, if $p_i < \check{p}_i$, it indicates that every customer will choose alternative i, provided it is within their choice set.
Additionally, we can refine these bounds to aim for target-specific outcomes, if desired. For instance, if p_i exceeds the m-th highest maximal breakpoint $\hat{p}_{i,m}^{nr}$, it suggests that at most m simulated customers will choose alternative i. Similarly, if p_i is below the m-th lowest minimal breakpoint $\check{p}_{i,m}^{nr}$, it implies that at least m simulated customers will choose alternative i. In our case, we assume that for each product, there should be at least one customer or scenario in which the product is chosen. If this assumption does not hold, the product can be considered as having no effect on the choice and can be removed from the set of offered products entirely. Consequently, we update the upper bound p_i^U by replacing it with $\hat{p}_i$ whenever $\hat{p}_i < p_i^U$, for all $i \in C^1$.

4.4 Node enumeration

Finally, when multiple nodes in the branch and bound tree have an identical upper bound, we can use the knowledge of a good solution to determine the next node to explore. Specifically, we select the node that contains the highest number of price values of the provided solution within its bounds, exploring promising areas first. Let Ω denote the set of all active nodes in the tree, where a node j consists of a set of bounds Δ^j and its upper bound on the objective value $\hat{o}_j$. Furthermore, denote p^* as the provided starting solution. The next node to be selected from the tree can now be written as:

$$j^* \coloneqq \arg\max_{j \in \Omega} \{|\{k \in C^1 \text{ s.t. } p_k^* \in [p_k^L(j), p_k^U(j)]\}| \text{ s.t. } \hat{o}_j = \max\{\hat{o}_i \mid \{\Delta^i,\ \hat{o}_i\} \in \Omega\}\}$$

implying that, among the nodes that reach the highest upper bound on their potential objective value, we choose the one that contains the highest number of price values of the initial solution in its bounds. In general, it is advisable to set p^* to the best incumbent solution identified up to that point.

5 Results and discussion

In this section, we apply the introduced methodology to a parking choice case study, evaluating its performance across various instance sizes. We explore the limits of solvable instances, compare the results to state-of-the-art methods for choice-based pricing and mixed logit-specific pricing, and assess the computational efficiency and solution quality.

5.1 Case study

To test the presented methodology we rely on the same mixed logit (ML) case study as various other studies, among which Bortolomiol et al. (2021); Paneque et al. (2022); Marandi & Lurkin (2023), establishing itself as a popular benchmark data set for ML-based pricing policies. The case study concerns a parking services operator, motivated by the published disaggregate demand model for parking choice by Ibeas et al. (2014). The choice set consists of three services: paid on-street parking (PSP), paid parking in an underground car park (PUP), and free on-street parking (FSP), presenting the opt-out. We artificially add more PSP or PUP options by duplicating the respective alternative and increasing the access time from the parking space to the desired destination by three minutes per duplicate, starting with duplicating the PUP alternative, and then the PSP alternative. For example: if we consider five offered alternatives, three of those will be PUP and two will be PSP. This extension corresponds to augmenting the parking space facilities in size and offering separate prices depending on proximity to the desired destination.

5.2 Description of experiments

The methods used in our experiments and their abbreviations are: Mixed-integer linear programming (MILP), Branch and Benders Decomposition (B&BD), B&BD without

Benders decomposition (B&B), Breakpoint Exact Algorithm (BEA), Breakpoint Heuristic Algorithm (BHA), Iterated Local Search (ILS), the state-of-the-art heuristic for the capacitated CPP, the Lagrangian decomposition approach presented in Paneque et al. (2022), which we will refer to as LAG, and finally the state-of-the-art exact method for ML-based pricing without capacities, the convexification of a biconvex optimization and trust-region algorithm (CoBiT), presented by Marandi & Lurkin (2023). The goal of our experiments is to answer the following set of questions:

1. How does the BEA, adapted to capacity constraints, compare in terms of the CPP instances we are able to solve to the state-of-the-art exact MILP approach when an exogenous priority queue is set in place?
2. How do the BHA and ILS compare to the MILP and BEA approaches on instances with capacity constraints and a priority queue?
3. What are the largest instances we can solve within 72 hours using the BHA with capacity constraints?
4. How do the BHA and ILS compare to the state-of-the-art exact B&BD and BEA approaches on pricing instances without capacity constraints?
5. How does the number of alternatives impact the optimality gap for the BHA?
6. What are the largest instances we can solve within 72 hours using the BHA without capacity constraints?
7. To what extent are we able to speed up the B&BD method using the solution from the BHA for guidance, together with the derived valid inequalities?
8. How do the proposed general solution methods compare to the mixed-logit specific LAG and CoBiT algorithms?

Table 0 – Summary of Tests

	Test 1	**Test 2**	**Test 3**	**Test 4**
J	2	2, 4	2, 4	2, 3, 4, 5, 6
N	50	50	50	50
R	2, 5, 10, 25, 50, 100, 250	2, 5, 10, 25, 50, 100, 200, 250	2, 5, 10, 25, 50, 100, 200, 250	1,000, 2,000, 3000
Capacities	[20, 20]	[20, 20], [15, 15, 15, 15]	[20, 20], [15, 15, 15, 15]	[20] $\forall i \in C_n$
Methods	MILP, BEA	BEA, BEA-M, BEA-R	MILP, BEA, BHA, ILS	BHA

	Test 5	**Test 6**	**Test 7**	**Test 8**
J	4	3, 4, 5, 6, 7, 8, 9, 10	2, 3, 4, 5, 6	2, 3, 4, 5, 6
N	20	20	50	10, 50, 100, 150, 197
R	100, 200, 300, 500, 1,000	20	500,000, 1,000,000	25, 50, 100, 200, 400
Capacities	[∞, ∞, ∞, ∞]	[∞] $\forall i \in C_n$	[∞] $\forall i \in C_n$	[20], [40], [60], [80], [∞] $\forall i \in C_n$
Methods	B&BD, BEA, BHA, ILS	B&BD, BHA	BHA	LAG, CoBiT, BEA, BEA, BHA, ILS, B&B, B&BD

To investigate these eight issues we perform the tests described in Table 0, where N denotes the number of randomly sampled individuals, R the number of scenarios generated and J the number of controlled alternatives. The limits for capacities are adapted from Paneque et al. (2021), as are the bounds for all prices, which are defined to be [0.5, 0.7] for PSP alternatives and [0.65, 0.85] for PUP alternatives. The initial starting point for the BHA in all cases is the mean of the bounds, i.e. $p_i^{start} = \frac{p_i^L + p_i^U}{2}$ $\forall i \in C$[1]. For the ILS, we use the following hyperparameter inputs: $\delta = 0.005, k = 3, \gamma = 2, \Delta_{max} = 0.05$. Both the MILP and B&BD experiments are performed using GUROBI 12.0.1 (Gurobi Opti-

mization, LLC, 2021). All methods are run on a single thread in a computational cluster node with two 2.4 GHz Intel Xeon Platinum 8360Y processors, where we utilize 16 cores with a total of 32 GB of RAM.

5.3 Numerical results and analysis

Table 1 – Test 1: MILP vs. BEA in the capacitated case

			MILP		BEA	
N	R	J	Time (s)	Revenue	Time (s)	Revenue
50	2	2	4.17	27.61	0.43	27.61
50	5	2	46.95	26.51	1.72	26.51
50	10	2	180.85	27.06	11.42	27.06
50	25	2	3,119.66	27.08	169.08	27.08
50	50	2	>5 hours	≥25.15	1,272.68	26.85
50	100	2	>25 hours	≥25.11	9,928.57	26.85
50	250	2	>45 hours	≥23.45	>45 hours	≥26.37

Table 1 presents the results of Test 1, comparing the exact MILP approach with the BEA extended to capacity constraints with a fixed priority queue. For small instances (up to $R = 25$), both methods were able to solve the instances within the time limit and achieved identical revenues, indicating that the BEA reliably recovers optimal solutions in these cases. As R increases, the MILP approach exceeds the time limit of 5 hours starting from $R = 50$, while the BEA continues to return feasible solutions within a reasonable timeframe. Notably, for larger instances ($R = 100$ and $R = 250$), the BEA produces higher revenues than the best solutions reported by the MILP before timeout, suggesting that the heuristic is able to identify high-quality solutions even when the exact method fails to converge.

Table 2 – Test 2: BHA and ILS vs. MILP and BEA in the capacitated case

			MILP		BEA		BHA		ILS	
N	R	J	Time (s)	Revenue	Time (s)	Revenue	Time (s)	Revenue	Time (s)	Revenue
50	2	2	4.17	27.61	0.43	27.61	0.22	27.61	1.03	27.61
50	5	2	46.95	26.51	1.72	26.51	0.32	26.46	5.91	26.51
50	10	2	180.85	27.06	11.42	27.06	0.58	27.05	20.34	27.06
50	25	2	3,119.66	27.08	169.08	27.08	3.40	27.05	129.66	27.08
50	50	2	>5 hours	≥25.15	1,272.68	26.85	8.31	26.53	559.04	26.85
50	100	2	>25 hours	≥25.11	9,928.57	26.85	51.77	26.72	2,791.28	26.85
50	250	2	>45 hours	≥23.45	>45 hours	≥26.37	455.37	26.66	15,867.67	26.71
50	10	4	>10 hours	≥22.21	>10 hours	≥25.10	7.08	26.78	527.34	26.83
50	50	4	>20 hours	≥22.19	>20 hours	≥25.19	166.21	27.00	7,234.88	27.00
50	100	4	>45 hours	≥20.50	>45 hours	≥26.09	866.97	26.67	34,050.57	26.67
50	200	4	>72 hours	≥20.32	>72 hours	≥24.79	2,762.39	26.70	106,286.13	26.70

Table 2 compares the performance of the MILP, BEA, BHA, and ILS approaches on capacitated instances with a fixed priority queue. For small instances ($R \leq 25$), all four methods are able to solve the problem within the time limit and achieve similar revenues, indicating that the heuristic methods match the performance of the exact methods in these cases. As the instance size increases, the MILP and BEA approaches begin to exceed the time limit, while both BHA and ILS continue to return high-quality solutions. Notably, for all instances up to $R = 250$ and $J = 2$, the BHA completes in under one hour, and its solutions are consistently close in value to those found by ILS, which itself aligns with the best-known solutions reported by the exact methods. In the larger four-price cases, neither MILP nor BEA solve any instance within the time limit, whereas BHA and ILS are able to return solutions throughout. In these settings, BHA requires significantly less computation time than ILS while still producing solutions of comparable quality.

Table 3 – Test 3: Limits of the BHA in the capacitated case

N	R	J	BHA (s)
50	1,000	2	15,093
50	1,000	3	25,326
50	1,000	4	69,134
50	1,000	5	112,042
50	1,000	6	178,923
50	2,000	2	51,637
50	2,000	3	84,231
50	2,000	4	150,132
50	2,000	5	193,233
50	3,000	2	164,922
50	3,000	3	184,293
50	3,000	4	>259,200

Table 3 explores the scalability of the BHA heuristic on increasingly large instances under capacity constraints. All instances were run with a time limit of 72 hours. The results show that the BHA successfully solves instances with up to 1,000 simulated draws and 6 offered prices, up to 2,000 draws for up to 5 prices, and up to 3,000 draws for up to 3 prices. The only configuration that exceeds the time limit corresponds to the most complex tested setting: 3,000 draws and 4 prices. These results demonstrate the ability of the BHA to handle significantly larger instances than previously reported in the literature, for example, compared to the maximal number of 200 draws for two offered alternatives in Paneque et al. (2022).

Table 4 – Test 4: BHA and ILS vs. B&BD and BEA in the uncapacitated case

			B&BD		BEA		BHA		ILS	
N	R	J	Time (s)	Revenue	Time (s)	Revenue	Time (s)	Revenue	Time (s)	Revenue
20	100	4	12,478	10.14	61,139	10.14	0.00	10.14	0.14	10.14
20	200	4	29,213	10.40	>24 hours	≥10.21	0.01	10.40	0.41	10.40
20	300	4	>24 hours	≥10.38	>24 hours	≥9.84	0.02	10.24	0.64	10.24
20	400	4	>24 hours	≥9.81	>24 hours	≥9.82	0.05	10.26	0.78	10.26
20	500	4	>24 hours	≥10.01	>24 hours	≥9.80	0.13	10.24	1.37	10.24

For the uncapacitated case, Haering et al. (2023) have shown that instances with two or fewer prices can easily be solved with the BEA algorithm, whereas for instances with

at least three alternatives, the B&BD approach is the fastest method. We thus consider only instances with at least three offered alternatives. Table 4 presents results comparing the performance of BHA and ILS to the exact B&BD and BEA approaches in the uncapacitated setting. For the smallest instance ($R = 100$), all methods converge within the time limit and produce identical revenues. At $R = 200$, the BEA fails to solve the instance within 24 hours, while both BHA and ILS match the solution obtained by B&BD. For larger instances ($R \geq 300$), neither B&BD nor BEA complete within the time limit, whereas BHA and ILS return feasible solutions in under two seconds and two minutes, respectively. In all unterminated cases, the heuristic methods produce solutions with higher objective values than those reported by the exact methods before timeout. BHA continues to match the performance of ILS with significantly shorter runtimes, and both heuristics recover the best known solution for all instances tested.

Table 5 – Test 5: BHA optimality gap when increasing dimensions

N	R	J	BHA	B&BD	Gap (%)
20	20	3	10.281	10.281	0
20	20	4	10.271	10.28	0.09
20	20	5	10.283	10.294	0.11
20	20	6	10.290	10.302	0.12
20	20	7	10.292	10.306	0.14
20	20	8	10.330	10.336	0.06
20	20	9	10.329	10.335	0.06
20	20	10	10.293	10.300	0.07

Table 5 depicts the outcomes of the analysis of the behavior of the optimality gap when the number of offered alternatives is increased. The instance size is kept small in order to be able to compute the global optimum with the B&BD algorithm within a reasonable time. We observe that generally, the mean of the bounds as a starting point does not lead to convergence to the global optimum, especially if the number of dimensions increases. However, the optimality gap is continuously very small and never reaches values bigger than 0.14%. We conclude that the BHA manages to deliver very high-quality solutions even for high-dimensional instances.

Table 6 – Test 6: Testing BHA limits without capacity constraints

N	R	J	BHA (s)
50	500,000	2	56
50	500,000	3	77
50	500,000	4	187
50	500,000	5	163
50	500,000	6	194
50	1,000,000	2	68
50	1,000,000	3	132
50	1,000,000	4	312
50	1,000,000	5	300
50	1,000,000	6	412

The next test, whose results are found in Table 6, aims to understand the limits in terms of instance size for the BHA when no capacity constraints are set in place. Although a 72 hour time limit was set in place, the largest instance, $N = 50, R = 1,000,000, J = 6$ was solved in less than seven minutes, demonstrating that the BHA is capable of tackling much larger instances than the ones available from our current dataset.

Table 7 – Test 7: B&BD with Guidance - 10% gap

N	R	J	normal w/out VIs (s)	normal w VIs (s)	Guided w/out VIs (s)	Guided w VIs (s)	Speedup from just VIs (%)	Add. Speedup from Sol. (%)	Total speedup (%)
50	1,000	3	987	1,132	731	816	-14.69	27.92	17.33
50	2,000	3	2,878	3,490	2,513	2,693	-21.26	22.84	6.43
50	3,500	3	10,325	12,919	6,390	7,454	-25.12	42.3	27.81
50	1,000	4	4,662	3,311	3,705	2,472	28.98	25.34	46.98
50	2,000	4	17,599	12,068	10,868	8,288	31.43	31.32	52.91
50	3,500	4	48,445	31,210	40,061	29,504	35.58	5.47	39.1
50	1,000	5	8,242	5,428	5,664	3,914	34.14	27.89	52.51
50	2,000	5	25,842	16,641	17,420	12,268	35.6	26.28	52.53
50	3,500	5	114,216	81,826	85,083	58,754	28.36	28.2	48.56

The outcomes of Test 7 are displayed over the next three tables, Tables 7, 8 and 9 respectively. They each depict the impact on computational time coming from adding the valid inequalities (VIs) to the McCormick relaxation in each node, the BHA solution as a starting point for the B&B search (referred to as "guided") and the two enhancements combined, for achieving different optimality gaps. Table 7 shows that to reach an optimality gap of 10%, the total speedup from the enhancements over the unmodified B&BD

reaches up to 50%, especially for instances with higher dimensions. Going further to 5% optimality in Table 8, the speedup reduces slightly and varies between 25-40%, decreasing further when looking at the computational time needed to reach 1% optimality gap in Table 9, which is reduced by 10-20% when using the enhancements. These speedups, where observable, remained stable until full convergence to the optimal solution (not depicted as only a fraction of these large instances were solved to optimality within the time limit). Two tendencies are clearly shown: For three alternatives, the VIs in fact seem to slow down the computation, however, increasing the number of alternatives shows that the more alternatives are being controlled, the larger the observed increase in speed when adding both the VIs or the heuristic starting point.

Table 8 – Test 7: B&BD with Guidance - 5% gap

N	R	J	normal w/out VIs (s)	normal w VIs (s)	Guided w/out VIs (s)	Guided w VIs (s)	Speedup from just VIs (%)	Add. Speedup from Sol. (%)	Total speedup (%)
50	1,000	3	2,372	2,454	1,933	2,245	-3.46	8.52	5.35
50	2,000	3	7,883	8,359	7,106	7,342	-6.04	12.17	6.86
50	3,500	3	51,964	57,229	42,991	47,282	-10.13	17.38	9.01
50	1,000	4	12,062	10,668	10,490	8,934	11.56	16.25	25.93
50	2,000	4	43,829	36,524	36,222	32,929	16.67	9.84	24.87
50	3,500	4	259,200	240,767	238,777	198,981	7.11	17.36	23.23
50	1,000	5	24,371	20,590	19,519	16,930	15.51	17.78	30.53
50	2,000	5	84,104	60,814	70,676	48,541	27.69	20.18	42.28
50	3,500	5	259,200	259,200	259,200	247,944	-	-	-

Table 9 – Test 7: B&BD with Guidance - 1% gap

N	R	J	normal w/out VIs (s)	normal w VIs (s)	Guided w/out VIs (s)	Guided w VIs (s)	Speedup from just VIs (%)	Add. Speedup from Sol. (%)	Total speedup (%)
50	1,000	3	15,840	16,933	13,239	14,594	-6.9	13.81	7.87
50	2,000	3	42,261	45,223	35,882	37,137	-7.01	17.88	12.12
50	3,500	3	183,696	195,743	152,833	162,594	-6.56	16.93	11.49
50	500	4	47,101	46,719	47,963	43,190	0.81	7.55	8.3
50	1,000	4	131,122	135,564	107,288	105,596	-3.39	22.11	19.47
50	1,500	4	229,620	230,187	203,348	202,560	-0.25	12	11.78
50	2,000	4	259,200	259,200	259,200	259,200	-	-	-
50	500	5	139,618	125,755	115,783	109,084	9.93	13.26	21.87
50	1,000	5	259,200	259,200	259,200	259,200	-	-	-

Table 10 – Test 8: Runtime (seconds) comparison to LAG

N	R	J	LAG	BEA	BHA	x Sp.	ILS	x Sp.
50	50	2	>7,200	3,109	24	300	841	9
50	100	2	>7,200	>7,200	96	75	3,640	2
50	200	2	>7,200	>7,200	459	16	>7,200	-
100	100	2	>7,200	>7,200	554	13	>7,200	-
150	100	2	>7,200	>7,200	1166	6	>7,200	-
197	100	2	>7,200	>7,200	1617	4	>7,200	-
50	25	4	>7,200	>7,200	31	230	2,711	3
50	50	4	>7,200	>7,200	148	49	7,157	1
50	100	4	>7,200	>7,200	591	12	>7,200	-

Regarding Test 8, we first compare our methods with the state-of-the-art heuristic method for general capacitated choice-based pricing, the LAG algorithm by Paneque et al. (2022). Unfortunately, the authors were not able to provide us with their code, which is why we replicate their experimental environment (12 threads on a 3.33 GHz Intel Xeon X5680 server running a 64-bit Ubuntu 16.04.2) in order to be able to compare with the runtimes they provide in their paper. In their experiments, they run tests on the same parking space operator case study (with capacity constraints) in the following way: a time limit of two hours is set and the achieved optimality gap after the time limit is reported, but no objective values or optimal prices. This makes the comparison to our approaches difficult, as neither the exact BEA nor the heuristic BHA and ILS procedures report an optimality gap. Table 10 shows the comparison between the LAG, the BEA, the BHA and the ILS in terms of runtimes.

Table 11 – Test 8: Objective value comparison to LAG (with a two-hour time limit)

			LAG	BEA		BHA		ILS
N	R	J	Gap (%)	Revenue	Gap* (%)	Revenue	Gap* (%)	Revenue
50	50	2	2.02	26.243	0.00	26.237	0.02	26.243
50	100	2	2.80	26.560	1.28	26.906	0.00	26.906
50	200	2	3.67	26.250	1.14	26.530	0.09	26.553
100	100	2	1.98	52.780	0.53	53.028	0.06	53.059
150	100	2	1.91	80.370	0.33	80.640	0.00	80.640
197	100	2	-	104.640	0.70	105.181	0.19	105.381
50	25	4	3.34	25.728	4.26	26.873	0.00	26.873
50	50	4	4.57	25.180	3.52	26.099	0.00	26.099
50	100	4	5.19	26.090	2.90	26.870	0.00	26.870

* relative gap compared to ILS revenue.

The time limit is set to two hours and the instance sizes and capacity limitations are chosen in accordance with the tests conducted in Paneque et al. (2022). The LAG does not solve any of the instances to optimality within the time limit, the BEA manages to solve the smallest one ($N = 50, R = 50, J = 2$), the BHA converges for all instances and the ILS does so for four out of the nine tested instances. The BHA on average is at least 78x faster than the LAG, with the ILS on average being at least 4x faster. These factors are likely many times higher in reality, since closing the last few percentages of the optimality gap notoriously takes very long, which was also reported to be the case by Paneque et al. (2022). Comparing the numerical results is difficult, since for the LAG, we only have the achieved optimality gap. Table 11 thus shows the reported optimality gap for the LAG, the achieved revenues for the BEA, BHA and ILS, as well as the relative gap between the revenues of the BEA and BHA compared to the ILS, as we cannot report the real optimality gap. We note that within this time limit, both heuristic procedures produce better solutions than the exact BEA approach, with the average relative gap between BEA and ILS being 1.63%, and the average relative gap between the BHA and ILS being 0.04%. This further demonstrates that, although the BHA is significantly more efficient in terms of computational time, there is only a small loss in terms of solution quality when compared to the ILS algorithm. The average reported optimality gap of the LAG is with 3.19% pronouncedly larger than the average relative gap from BHA to ILS, however, the comparison should be contextualized carefully, considering the lack of an optimality certificate for the ILS.

Table 12 – Test 8: Runtime (seconds) comparison to CoBiT

N	n^2	R	J	CoBiT	B&B	x Sp.	B&BD	x Sp.	BEA	x Sp.	BHA	x Sp.	ILS	x Sp.
10	9	100	2	69	17	4	83	0.83	1	69	0.002	$4 \cdot 10^4$	0.086	971
10	9	100	3	607	124	5	623	0.97	10	61	0.001	$4.5 \cdot 10^5$	0.117	5,343
10	9	100	4	6,439	985	7	4,791	1.34	5727	1	0.002	$3.2 \cdot 10^6$	0.216	$2.2 \cdot 10^4$
10	9	100	5	34,409	4,017	9	18,644	1.85	>86,400	-	0.001	$27 \cdot 10^6$	0.250	$7.4 \cdot 10^4$
10	9	100	6	39,164	6,015	7	27,758	1.41	>86,400	-	0.002	$15 \cdot 10^6$	0.254	$1.1 \cdot 10^5$
10	64	400	2	270	128	2	620	0.44	1	270	0.003	$8.6 \cdot 10^4$	0.139	4,461
10	64	400	3	4,234	783	5	3,174	1.33	560	8	0.005	$9.2 \cdot 10^5$	0.244	$1.2 \cdot 10^4$
10	64	400	4	37,384	8,895	4	34,503	1.08	>86,400	-	0.010	$3.8 \cdot 10^6$	0.460	$7.5 \cdot 10^4$
10	64	400	5	38,090	25,367	2	>86,400	-	>86,400	-	0.010	$3.6 \cdot 10^6$	0.658	$1.3 \cdot 10^5$
10	64	400	6	39,424	>86,400	-	>86,400	-	>86,400	-	0.012	$3.4 \cdot 10^6$	0.650	$1.3 \cdot 10^5$

We next compare to the state-of-the-art algorithm for (uncapacitated) mixed-logit-basing pricing, the CoBiT algorithm presented by Marandi & Lurkin (2023). The authors use the same parking choice case study as we do to illustrate their algorithm, and they have made their code available on GitHub. In their experiments, they address cases with $N = 10$ customer classes and two controlled prices, achieving optimal solutions. They approximate the continuous distributions of price sensitivity and arrival times with discrete distribu-

tions and report the runtimes based on varying numbers of breakpoints in their approximations.

Differing from our approach, their model assigns each pair of customer class and breakpoint to a customer under the MNL model, leading to nonlinear choice probabilities with respect to prices. Our method simulates each customer with deterministic preferences based on scenario draws. As a result, the problem we address for a given (N, R) differs and may not approximate the original model in the same way as their model with identical N and $R = n^2$ (number of breakpoints). According to numerical experiments conducted in Haering et al. (2023), $R = 400$ draws in sample average simulation achieve similar accuracy to $n^2 = 64$ in Marandi & Lurkin (2023). Therefore, we consider good approximations with $n^2 = 64$ or $R = 400$, and smaller instances with $n^2 = 9$ or $R = 100$. We employ the same 10 customer classes as used in Marandi & Lurkin (2023), utilizing Julia version 1.8.0 and a computational cluster node with identical specifications to our other experiments, with all price bounds set to the closed unit interval [0, 1] as per the authors' specifications, as well as no limitations on capacity. To ensure accurate runtime comparisons, we rerun the authors' method and successfully replicate their results. Additionally, we increase the number of controlled alternatives to up to six, as described in our previous experiments, to assess the impact of a larger choice set.

Table 12 reports the runtimes of all methods on instances solvable by CoBiT, an exact method specifically tailored to mixed-logit pricing. In contrast to the specialized approach, the BHA and ILS—despite being general-purpose heuristics—consistently solve all instances within a fraction of the time. While CoBiT exceeds 24 hours on several of the more complex cases ($J \geq 4$), the BHA returns solutions in under a second, and the ILS does so within two minutes. On all instances where the optimal value is known (i.e., when CoBiT terminates), the ILS recovers the same revenue, while the BHA's average deviation remains within 0.02%. These results indicate that the proposed heuristics are effective not only in terms of computational efficiency but also in recovering near-optimal solutions even in the specific mixed-logit context for which CoBiT is designed. Numerical results are detailed in the Appendix in Tables A1, A2, A3 and A4. It is important to mention a difference in how the price sensitivity parameters are generated between their framework and ours: in order to ensure behavioral realism, we opt to draw the factors β_p^{in} multiplying the prices using a truncated normal distribution, to guarantee that they are always negative. This additional step in the simulation procedure was not performed by Marandi & Lurkin (2023). Furthermore, the size of the confidence set for evaluating the integral (they chose a 99% set) influences the numerical value of the obtained revenue. It is for these reasons that there is a discrepancy in the optimal values for the prices and the objective between our approaches and CoBiT.

5.4 Summary of results

We summarize our findings as follows: The BEA, adapted to capacity constraints, is able to solve a broader set of capacitated CPP instances than the MILP formulation within the time limit, achieving identical revenues on all completed instances and returning better solutions in cases where the MILP times out ($R = 250$). The BHA and ILS heuristics show strong computational performance in these settings: while ILS consistently finds globally optimal solutions, the BHA returns near-optimal solutions with an average deviation below 0.2%, and both methods successfully solve instances that are out of reach for exact approaches when $R \geq 50$.
The BHA scales well to large capacitated instances, handling up to 1,000 simulation draws with 6 prices and up to 3,000 draws for 3 prices within the 72-hour limit. In the uncapacitated case, BHA and ILS solve all tested instances quickly, even when exact methods such as B&BD and BEA fail to terminate. Both heuristics match the best-known solutions in all cases where verification is possible, with BHA again exhibiting notably shorter runtimes. The BHA also contributes to improving the exact B&BD method: when used to initialize the tree and generate valid inequalities, it reduces solution time by up to 50%, depending on the tightness of the optimality gap. Compared to specialized state-of-the-art approaches, the BHA significantly outperforms the heuristic LAG algorithm in the capacitated setting, achieving at least a 78-fold reduction in runtime. In mixed-logit pricing problems, BHA and ILS also solve all benchmark instances significantly faster than the exact CoBiT approach, with ILS always matching the optimal value and BHA showing an average deviation below 0.02%.
Overall, the proposed heuristics demonstrate strong empirical performance across capacitated and uncapacitated variants of the CPP, providing fast and high-quality solutions even in high-dimensional and computationally challenging scenarios.

6 Conclusions

This research introduces the Breakpoint Heuristic Algorithm (BHA), a scalable and efficient approach for solving the choice-based pricing problem (CPP), both with and without capacity constraints. We also propose an extension of the Breakpoint Exact Algorithm (BEA) to handle capacities through a priority-queue-based strategy, and we enhance the state-of-the-art Branch-and-Benders Decomposition (B&BD) method by incorporating valid inequalities that make use of a strong incumbent solution, as for example retrieved by a heuristic.
The adapted BEA is shown to solve a broader set of capacitated instances than the existing MILP formulation within a fixed time limit, offering better or equal revenue in all tested cases. The BHA, based on a coordinate descent scheme, performs particularly well in high-dimensional settings and solves many large-scale instances that are intractable for exact approaches within practical timeframes. Compared to the LAG algorithm for capacitated pricing, the BHA delivers substantial reductions in computation time while maintaining high solution quality. In the uncapacitated case, the BHA consistently matches the

best-known solutions and, when used to warm-start the B&BD algorithm, leads to computational gains of up to 50%. The ILS extension of the BHA succeeds in recovering the global optimum across all verifiable instances, although at the cost of increased runtime. Finally, in mixed-logit pricing problems, the BHA and ILS outperform CoBiT, a method specifically designed for such models. Both heuristics solve all tested instances substantially faster, with ILS consistently matching the optimal value and BHA maintaining an average optimality gap below 0.02%. Owing to its flexibility, generalizability, and strong empirical performance, the BHA constitutes a promising tool for large-scale pricing applications involving choice models.

We thus conclude that we have successfully contributed to filling the gaps identified in the literature: we provide operational algorithms to solve the CPP, capable of handling large, high-dimensional instance sizes and complex additional constraints (like capacity), while maintaining only weak assumptions, if any, on the choice model.

Various avenues are open for future research: In terms of the heuristic, other extensions of the BHA to escape local optima should be considered, as the ILS increases computational time substantially. The remarkable speed of the BHA algorithm, together with its capability to produce high-quality solutions and overall flexibility due to depending only on evaluating an objective function given fixed parameters, lays the groundwork for its application in larger as well as more intricate problem settings.

References

Abdolhamidi, D., & Lurkin, V. (2024). A tactical time slot management problem under mixed logit demand. *arXiv preprint arXiv:2407.02308*.

Benati, S., & Hansen, P. (2002). The maximum capture problem with random utilities: Problem formulation and algorithms. *European Journal of operational research*, *143*(3), 518–530.

Bortolomiol, S., Lurkin, V., & Bierlaire, M. (2021). A simulation-based heuristic to find approximate equilibria with disaggregate demand models. *Transportation Science*, *55*(5), 1025–1045.

Cordone, R., & Redaelli, F. (2011). Optimizing the demand captured by a railway system with a regular timetable. *Transportation Research Part B: Methodological*, *45*(2), 430–446.

Davis, J. M., Topaloglu, H., & Williamson, D. P. (2017). Pricing problems under the nested logit model with a quality consistency constraint. *INFORMS Journal on Computing*, *29*(1), 54–76.

Gallego, G., & Wang, R. (2014). Multiproduct price optimization and competition under the nested logit model with product-differentiated price sensitivities. *Operations Research*, *62*(2), 450–461.

Gilbert, F., Marcotte, P., & Savard, G. (2014). Mixed-logit network pricing. *Computational Optimization and Applications*, *57*, 105–127.

Gosavi, A. (2015). *Simulation-based optimization* (Vol. 55). doi: 10.1007/978-1-4899-7491-4

Gurobi Optimization, LLC. (2021). *Gurobi Optimizer Reference Manual.* Retrieved from `https://www.gurobi.com`

Haase, K., & Müller, S. (2013). Management of school locations allowing for free school choice. *Omega*, *41*(5), 847–855.

Haering, T., Legault, R., Torres, F. A., Ljubic, I., & Bierlaire, M. (2023). Exact algorithms for continuous pricing with advanced discrete choice demand models. *OR Spectrum*. (Accepted for publication)

Hanson, W., & Martin, K. (1996). Optimizing multinomial logit profit functions. *Management Science*, *42*(7), 992–1003.

Ibeas, A., Dell'Olio, L., Bordagaray, M., & Ortúzar, J. d. D. (2014). Modelling parking choices considering user heterogeneity. *Transportation Research Part A: Policy and Practice*, *70*, 41–49.

Korfmann, F. (2018). *Essays on advanced discrete choice applications* (Unpublished doctoral dissertation). Staats-und Universitätsbibliothek Hamburg Carl von Ossietzky.

Legault, R., & Frejinger, E. (2024). A model-free approach for solving choice-based competitive facility location problems using simulation and submodularity. *INFORMS Journal on Computing*.

Li, H., Webster, S., Mason, N., & Kempf, K. (2019). Product-line pricing under discrete mixed multinomial logit demand: winner—2017 msom practice-based research competition. *Manufacturing & Service Operations Management*, *21*(1), 14–28.

Liberti, L. (2008). Introduction to global optimization. *Ecole Polytechnique*.

Liu, N., Ma, Y., & Topaloglu, H. (2020). Assortment optimization under the multinomial logit model with sequential offerings. *INFORMS Journal on Computing*, *32*(3), 835–853.

Ljubić, I., & Moreno, E. (2018). Outer approximation and submodular cuts for maximum capture facility location problems with random utilities. *European Journal of Operational Research*, *266*(1), 46–56.

Mai, T., & Lodi, A. (2020). A multicut outer-approximation approach for competitive facility location under random utilities. *European Journal of Operational Research*, *284*(3), 874–881.

Marandi, A., & Lurkin, V. (2023). An exact algorithm for the static pricing problem under discrete mixed logit demand. *EURO Journal on Computational Optimization*, *11*, 100073.

McCormick, G. P. (1976). Computability of global solutions to factorable nonconvex programs: Part i—convex underestimating problems. *Mathematical programming*, *10*(1), 147–175.

Müller, D., Nesterov, Y., & Shikhman, V. (2021). Dynamic pricing under nested logit demand. *arXiv preprint arXiv:2101.04486*.

Paneque, M. P., Bierlaire, M., Gendron, B., & Azadeh, S. S. (2021). Integrating advanced discrete choice models in mixed integer linear optimization. *Transportation Research Part B: Methodological*, *146*, 26–49.

Paneque, M. P., Gendron, B., Azadeh, S. S., & Bierlaire, M. (2022). A lagrangian decomposition scheme for choice-based optimization. *Computers & Operations Research*, *148*, 105985.

Rahmaniani, R., Crainic, T. G., Gendreau, M., & Rei, W. (2017). The Benders decomposition algorithm: A literature review. *European Journal of Operational Research*, *259*(3), 801–817.

Robenek, T., Azadeh, S. S., Maknoon, Y., de Lapparent, M., & Bierlaire, M. (2018). Train timetable design under elastic passenger demand. *Transportation research Part b: methodological*, *111*, 19–38.

Rusmevichientong, P., Shen, Z.-J. M., & Shmoys, D. B. (2010). Dynamic assortment optimization with a multinomial logit choice model and capacity constraint. *Operations research*, *58*(6), 1666–1680.

Shen, Z.-J. M., & Su, X. (2007). Customer behavior modeling in revenue management and auctions: A review and new research opportunities. *Production and operations management*, *16*(6), 713–728.

Sumida, M., Gallego, G., Rusmevichientong, P., Topaloglu, H., & Davis, J. (2021). Revenue-utility tradeoff in assortment optimization under the multinomial logit model with totally unimodular constraints. *Management Science*, *67*(5), 2845–2869.

van de Geer, R., & den Boer, A. V. (2022). Price optimization under the finite-mixture logit model. *Management Science*, *68*(10), 7480–7496.

Wu, D., Yin, Y., Lawphongpanich, S., & Yang, H. (2012). Design of more equitable congestion pricing and tradable credit schemes for multimodal transportation networks. *Transportation Research Part B: Methodological*, *46*(9), 1273–1287.

Appendix A Comparison to CoBiT

This section shows the numerical results for the CoBiT, B&B, BHA and ILS algorithms when applied to the artificially augmented parking choice data set.

Table A1 – Numerical results for CoBiT

N	R	J	Obj.	p_1	p_2	p_3	p_4	p_5	p_6
10	9	2	6.837	0.520	0.729				
10	9	3	6.761	0.500	0.690	1.000			
10	9	4	6.882	0.531	0.719	0.750	0.520		
10	9	5	6.842	0.540	0.722	0.750	0.500	0.697	
10	9	6	6.839	0.540	0.720	0.750	0.500	0.950	0.500
10	64	2	5.069	0.500	0.661				
10	64	3	5.080	0.500	0.659	0.664			
10	64	4	5.086	0.500	0.659	0.664	0.500		
10	64	5	5.084	0.500	0.662	0.660	0.500	0.625	
10	64	6	5.086	0.500	0.661	0.661	0.500	0.628	0.498

Table A2 – Numerical results for best exact method (B&B)

N	R	J	Obj.	p_1	p_2	p_3	p_4	p_5	p_6
10	100	2	5.200	0.626	0.651				
10	100	3	5.142	0.562	0.652	0.674			
10	100	4	5.140	0.543	0.560	0.652	0.677		
10	100	5	5.142	0.543	0.560	0.652	0.677	0.668	
10	100	6	5.155	0.540	0.575	0.530	0.652	0.656	0.680
10	400	2	5.279	0.550	0.652				
10	400	3	5.196	0.550	0.652	0.651			
10	400	4	5.204	0.549	0.570	0.652	0.655		
10	400	5	5.208	0.549	0.570	0.652	0.655	0.650	
10	400	6	5.235	0.564	0.556	0.538	0.650	0.661	0.669

Table A3 – Numerical results for BHA

N	R	J	Obj.	Gap (%)	p_1	p_2	p_3	p_4	p_5	p_6
10	100	2	5.200	0.00	0.626	0.651				
10	100	3	5.141	0.02	0.548	0.653	0.673			
10	100	4	5.140	0.00	0.543	0.560	0.652	0.677		
10	100	5	5.142	0.00	0.540	0.561	0.651	0.673	0.665	
10	100	6	5.152	0.05	0.543	0.579	0.654	0.653	0.657	0.681
10	400	2	5.279	0.00	0.550	0.652				
10	400	3	5.196	0.00	0.550	0.653	0.651			
10	400	4	5.201	0.06	0.532	0.534	0.650	0.658		
10	400	5	5.205	0.07	0.532	0.534	0.650	0.658	0.650	
10	400	6	5.234	0.03	0.564	0.558	0.542	0.650	0.661	0.669

Table A4 – Numerical results for ILS

N	R	J	Obj.	Gap (%)	p_1	p_2	p_3	p_4	p_5	p_6
10	100	2	5.200	0.00	0.626	0.651				
10	100	3	5.142	0.00	0.562	0.652	0.674			
10	100	4	5.140	0.00	0.543	0.560	0.652	0.677		
10	100	5	5.143	0.00	0.543	0.560	0.652	0.677	0.668	
10	100	6	5.155	0.00	0.540	0.576	0.530	0.652	0.656	0.680
10	400	2	5.279	0.00	0.550	0.652				
10	400	3	5.196	0.00	0.550	0.652	0.651			
10	400	4	5.204	0.00	0.549	0.569	0.652	0.657		
10	400	5	5.208	0.00	0.549	0.569	0.652	0.657	0.650	
10	400	6	5.234	0.03	0.564	0.558	0.542	0.650	0.661	0.669